Notes

330en13

13

BASIC CONCEPTS OF OOP

In the previous lesson you have learnt about the basics of C++ programming. Now you will learn about basic concepts of Object Oriented Programming (OOP). The object-oriented programming (OOP) is a different approach to programming and quite suitable for managing large and complex programs. An object oriented language combines the data to its function or code in such a way that access to data is allowed only through its function or code. In this lesson, you will learn about the various benefits provided by OOP and applications of OOP.

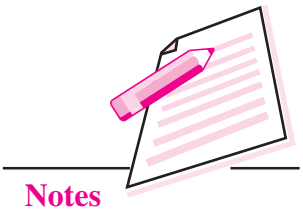
OBJECTIVES

After reading this lesson, you will be able to:

- learn the basic concepts used in OOP;
- describe the various benefits provided by OOP;
- explain the programming applications of OOP.

13.1 OBJECT ORIENTED PROGRAMMING

The object-oriented programming is a different approach to programming. It has been created with a view to increase programmer's productivity by overcoming the weaknesses found in procedural programming approach. Over the years many object-oriented programming languages such as C++, Java have come up and are becoming quite popular in the market. The major need for developing such languages was to manage the ever-increasing size and complexity of programs.



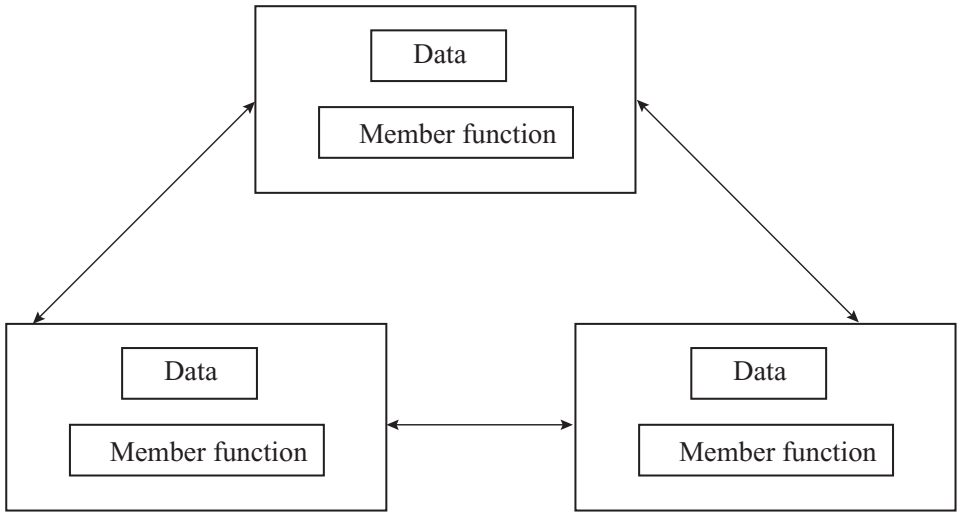


Fig: 13.1: Paradigm of OOP

13.1.1 Features of OOPS

The following are the features of object-oriented programming.

- Objects
- Classes
- Data abstraction
- Data encapsulation
- Inheritance
- Polymorphism

Objects

Object is a class variable or an instance of class. It can represent a person, a bank account or any item that a program can handle. When a program is executed, the objects interact by sending messages to one another.

For example, if 'customer' and 'account' are two objects in a program, then the customer object may send message to account object requesting for a bank balance. Each object contains data and code to manipulate data. Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted and the type of response returned by the objects.

Class

We have just mentioned that objects contain data and function or code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact objects are variables of type class. Once a class has been defined, we can create any number of objects associated with that class.

Class Name
Class variables
Class functions

Fig: 13.2: Class Structure

For example, there is a class Employee which has Sue, Bill, Al, Hal, David different employees. Each employee will have unique identity; so they will form the objects of the class Employee.

Class is a user defined data type. It is a blueprint of data and member functions.

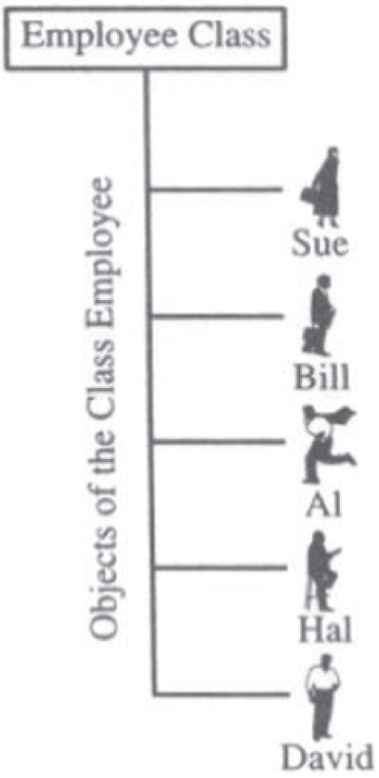


Fig: 13.3: Employee class example

Data Abstraction

Abstraction refers to the act of representing essential features without including the background details. To understand this concept more clearly, take an example of 'switch board'. You only press particular switches as per your requirement. You need not know the internal working of these switches. What is happening inside is hidden from you. This is abstraction, where you only know the essential things to operate on switch board without knowing the background details of switch board.

Data Encapsulation

Wrapping up of data and functions into a single unit is called as data encapsulation. Encapsulation is the most basic concept of OOP. It is the way of combining both data and the functions that operate on that data under a single unit. The only way to access the data is provided by the functions (that are combined along with the data). These functions are considered as member functions in C++. It is not possible to access the data directly. If you want to reach the data item in an object, you call a member function in the object. It will read the data item and return the value to you. The data is hidden, so it is considered as safe and far away from accidental alternation. Data and its functions are said to be encapsulated into a single entity.

In the Figure 13.4 item is a class which has keep_data as member variable which cannot be accessed from outside directly. It can be accessed only via the member functions set() and get_value().

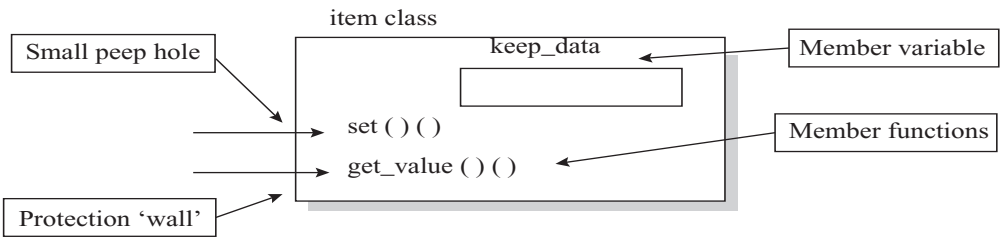


Fig: 13.4: Encapsulated Data and Functions in class item

Modularity

The act of partitioning a program into individual components is called modularity. It gives the following benefits.

- It reduces its complexity to some extent.
- It creates a number of well-defined, documented boundaries within the program.

Module is a separate unit in itself. It can be complied independently though it has links with other modules. Modules work quite closely in order to achieve the program's goal.

Inheritance

It is a process by which object of one class inherit the properties of objects of another class. It is the capability to define a new class in terms of an existing class. An existing class is known as a **base class** and the new class is known as **derived class**. Number of examples can be given on this aspect. For example, a motor cycle is a class in itself. It is also a member of two wheelers class. Two wheelers class in turn is a member of automotive class as shown in Fig. 13.5. The automotive is an example of base class and two wheelers is its derived class. In simple words, we can say a motor cycle is a two wheeler automotive.

C++ supports such hierarchical classification of classes. The main benefit from inheritance is that we can build a generic base class, and obtain a new class by adding some new features to an existing class and so on. Every new class defined in that way consists of features of both the classes. Inheritance allows existing classes to be adapted to new application without the need for modification.

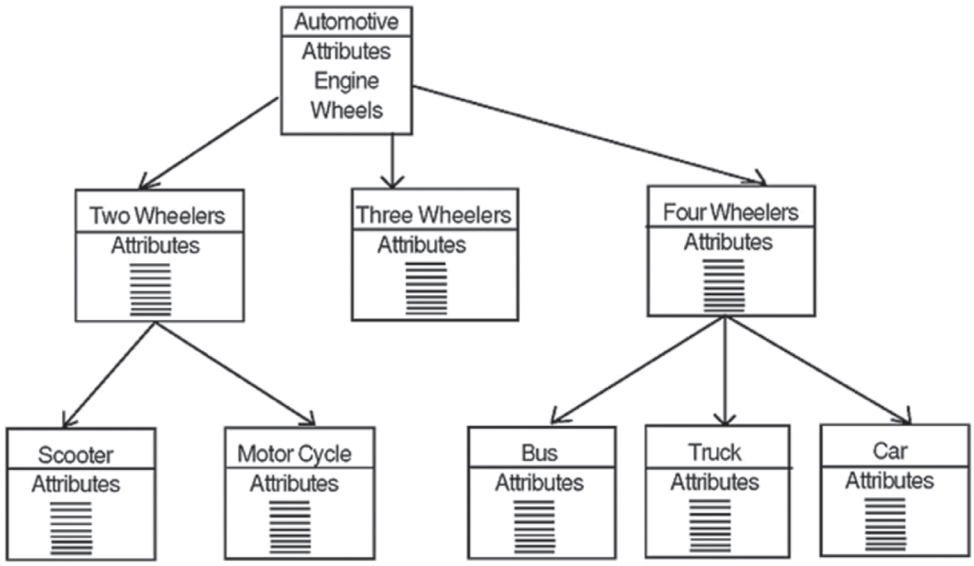


Fig: 13.5: Automotive class

Polymorphism

Polymorphism is a key to the power of OOP. It is the concept that supports the capability of data to be processed in more than one form. For example, an operation may exhibit different behavior in different instances. The behavior depends upon the types of data used in the operation.

For example let us consider Fig 13.6 the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings then the operation would produce a third string by concatenation.

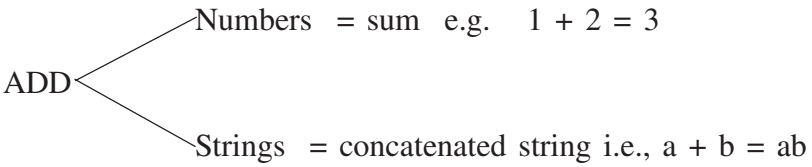
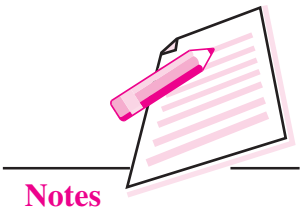


Fig: 13.6: Addition operation



INTEXT QUESTIONS 13.1

1. State whether the following statements are true or false:
   (a) In procedure-oriented programming all data is shared by all functions.
   (b) One of the striking features of OOP is division of program into objects that represent real world entities.
   (c) Wrapping up of data of different types into a single unit is known as encapsulation.
   (d) Object oriented programs are executed much faster than conventional programs.
   (e) Since C is a subset of C++, C programs will run under C++ compilers.

13.2 BENEFITS OF OOP

OOP provides lot of benefits to both the program designer and the user. Object-oriented approach helps in solving many problems related to software development and quality of software product. The new technology assures greater programmer productivity, better quality of software and lesser maintenance cost. The major benefits are:

- Software complexity can be easily managed.
- Object-oriented systems can be easily upgraded.
- It is quite easy to partition the work in a project based on objects.
- Objects created for object oriented program can easily be reused in other programs.

Programming Applications of OOP

OOP has become one of the programming buzzwords today. There appears to be a great deal of excitement and interest among software programmers in using OOP. Applications of OOP are gaining importance in many areas. OOP has been extensively used in the development of Windows and word based systems such as MS-Windows, x-Windows etc. The promising application areas of OOP are:

(i) **Multiple uses of data structure:** This is an application where the same data structure is used many times. For example a window data structure is used multiple-times in a windowing system.

(ii) **Data in multiple programs:** This is an application where the same operations are performed on a data structure in different programs. For example, record validation in an accounting system.

The other application areas of OOP are parallel programming, simulation and modeling, AI and Expert systems, Neural Networks and CAD systems, object-oriented databases.

INTEXT QUESTIONS 13.2

1. Fill in the blanks.
   (a) C++ is an ..... language.
   (b) An ..... is a self contained unit of ..... and function.
   (c) A ..... can be used to create objects of its own type.
   (d) ..... is a way of combining data with functions into an object.
   (e) A derived class can be derived from a ..... class.
2. State whether the following statements are true or false:
   (a) C++ is an extension to C programming language.
   (b) Functions of one object cannot access the functions of other objects in C++
   (c) Inheritance is the capability to define a new class in terms of an existing class.
   (d) Abstraction refers to the act of representing essential features without including the background details.
   (e) An object is a new data type.

WHAT YOU HAVE LEARNT

- Object is a class variable or an instance of a class.
- Abstraction refers to the act of representing essential features without including the background details.
- Data encapsulation is the way of combining both data and the functions that operate on that data under a single unit.

- Inheritance is a process by which object of one class inherit, the properties of objects of another class.
- Polymorphism means multiple forms of same thing.

TERMINAL EXERCISE

1. What are the features of OOP?
2. Explain the following terms briefly:
   (a) Data abstraction
   (b) Data encapsulation
   (c) Polymorphism
   (d) Inheritance
3. Describe the various benefits of OOP.

ANSWERS TO INTEXT QUESTIONS

13.1

1. (a) False (b) True (c) False
   (d) False (e) True

13.2

1. (a) OOP (b) Object, data (c) class
   (d) encapsulation (e) base
2. (a) True (b) False (c) True
   (d) True (e) False